In the Claims:

Please cancel claims 1-46

Please add new claims 47-109

47. (new) A compute node capable of operating as part of a distributed system, comprising:

memory; and

a processor configured to access the memory to perform a process in a distributed computation running on the distributed system, the processor being further configured to record a first set of memory locations modified by the processor during a first checkpoint interval, and create a checkpoint from the contents of the first set of memory locations, while recording a second set of memory locations modified by the processor during a second checkpoint interval.

48. (new) The compute node of claim 47 wherein the processor is further configured to write protect the first set of memory locations before modifying the second set of memory locations.

49. (new) The compute node of claim 48 wherein the processor is further configured to suspend the process between the first and second checkpoint intervals, the processor being further configured to write protect the first set of memory locations while the process is suspended.

50. (new) The compute node of claim 49 wherein the processor is further configured to enter a barrier following the completion of write protecting the first set of memory locations and exit the barrier before resuming the process during the second checkpoint interval.

51. (new) The compute node of claim 48 wherein the processor is further configured to create the checkpoint by storing the contents of the first set of memory locations, the processor being further configured to remove the write protection for a memory location from the first set when the processor needs to modify the memory

location during the second checkpoint interval after the contents of the memory location has been stored.

52. (new) The compute node of claim 51 wherein the processor is further configured to create the checkpoint by storing the contents of the first set of memory locations in a certain order, the processor further being configured to store the contents of a memory location from the first set earlier than it would otherwise be stored when the processor needs to modify the memory location during the second checkpoint interval.

53. (new) The compute node of claim 52 wherein the processor is further configured to remove the write protection for the memory location after the contents of the memory location has been stored.

54. (new) The compute node of claim 47 further comprising a checkpoint file, wherein the processor is further configured to create the checkpoint by storing the contents of the first set of memory locations to the checkpoint file.

55. (new) The compute node of claim 54 wherein the processor is further configured to remove the record of a memory location from the first set after the contents from the memory location is stored in the checkpoint file.

56. (new) The compute node of claim 47 wherein the processor is further configured to configured to create the checkpoint by storing the contents of the first set of memory locations to non-volatile storage.

57. (new) The compute node of claim 47 wherein the processor is further configured to store in the memory a copy of each message output from the compute node during the process until an acknowledgement is received, and output each message copied in the memory that does not receive an acknowledgement, and wherein the processor is further configured to receive messages during the process, and output an acknowledgement for each message received, the processor being further configured to

recognize and discard duplicate messages received by the compute node, and for each duplicate message, output an acknowledgement.

58. (new) A compute node capable of operating as part of a distributed system, comprising:

memory; and

a processor configured to perform a process in a distributed computation running on the distributed system, the processor being further configured to store in the memory a copy of each message output from the compute node until an acknowledgement is received, the processor being further configured to create a checkpoint, and if a subsequent failure occurs, roll the compute node back to the checkpoint and output each message copied in the memory that does not receive an acknowledgement after the compute node is rolled back to the checkpoint.

59. (new) The compute node of claim 58 wherein the processor is further configured to receive messages, and output an acknowledgement for each message received, the processor being further configured to recognize and discard duplicate messages received by the compute node, and for each duplicate message, output an acknowledgement.

60. (new) A compute node capable of operating as part of a distributed system, comprising:

a processor configured to perform a process in a distributed computation running on the distributed system, the processor being further configured to receive messages, and output an acknowledgement for each message received, the processor being further configured to create a checkpoint, and if a subsequent failure occurs, roll the compute node back to the checkpoint, recognize and discard duplicate messages received by the compute node after the compute node is rolled back to the checkpoint, and for each duplicate message, output an acknowledgement.

61. (new) The compute node of claim 60 further comprising memory, wherein the processor is further configured to store in the memory a copy of each message output from the compute node until an acknowledgement is received, the processor being further configured to output each message copied in the memory that does not receive an acknowledgement.

62. (new) A compute node capable of operating as part of a distributed system, comprising:

a processor configured to perform a process in a distributed computation running on the distributed system, the processor being further configured to create a checkpoint for the process, and in response to a preemptive scheduling request, store the checkpoint to non-volatile memory and halt the process.

63. (new) The compute node of claim 62 wherein the processor is further configured to use the stored checkpoint to resume the process.

64. (new) The compute node of claim 62 wherein the processor is further configured to perform a process in a second distributed computation running on the distributed system after the previous process is halted.

65. (new) The compute node of claim 62 wherein the processor is further configured to perform a second process in the distributed computation previously being performed by another compute node in the distributed system, the processor being further configured to resume the second process from the checkpoint last taken by said another compute node for the second process.

66. (new) Computer readable media embodying a program of instructions executable by a processor to perform a method of creating a checkpoint for a process in a distributed computation running on a distributed system, the method comprising:

recording a first set of memory locations modified by the process during a first checkpoint interval; and creating a checkpoint from the contents of the first set of memory locations, while recording a second set of memory locations modified by the process during a second checkpoint interval.

67. (new) The computer readable media of claim 66 wherein the method further comprises write protecting the first set of memory locations before the process modifies the second set of memory locations.

68. (new) The computer readable media of claim 67 wherein the method further comprises suspending the process between the first and second checkpoint intervals, and wherein the first set of memory locations are write protected while the process is suspended.

69. (new) The computer readable media of claim 67 wherein the method further comprises entering a barrier following the completion of write protecting the first set of memory locations and exiting the barrier before resuming the process during the second checkpoint interval.

70. (new) The computer readable media of claim 67 wherein the checkpoint is created by storing the contents of the first set of memory locations, the method further comprising removing the write protection for a memory location from the first set when the process needs to modify the memory location during the second checkpoint interval after the contents of the memory location has been stored.

71. (new) The computer readable media of claim 70 wherein the checkpoint is created by storing the contents of the first set of memory locations in a certain order, the method further comprising storing the contents of a memory location from the first set earlier than it would otherwise be stored when the process needs to modify the memory location during the second checkpoint interval.

72. (new) The computer readable media of claim 71 wherein the method further comprises removing the record of the memory location and the write protection for the memory location, after the contents of the memory location has been stored.

73. (new) The computer readable media of claim 66 wherein the checkpoint is created by storing the contents of the first set of memory locations to a checkpoint file.

74. (new) The computer readable media of claim 73 wherein the method further comprises removing the record of a memory location from the first set after the contents from the memory location is stored in the checkpoint file.

75. (new) The computer readable media of claim 66 wherein the checkpoint is created by storing the contents of the first set of memory locations to non-volatile storage.

76. (new) The computer readable media of claim 66 wherein the process is performed by a compute node in the distributed system, the method further comprising storing in the memory a copy of each message output from the compute node during the process until an acknowledgement is received, and outputting each message copied in the memory that does not receive an acknowledgement, and wherein the method further comprises receiving messages during the process, outputting an acknowledgement for each message received, recognizing and discarding duplicate messages received by the compute node, and for each duplicate message, outputting an acknowledgement.

77. (new) Computer readable media embodying a program of instructions executable by a processor to perform a method of creating a checkpoint for a process in a distributed computation running on a distributed system, the process being performed by a compute node, the method comprising:

storing in a memory a copy of each message output from the compute node until an acknowledgement is received;

creating a checkpoint for the process;

rolling compute node back to the checkpoint in response to a failure; and

outputting each message copied in the memory that does not receive an acknowledgement after the compute node is rolled back to the checkpoint.

78. (new) The computer readable media of claim 77 wherein the method further comprises outputting an acknowledgement for each message received by the compute node, and recognizing and discarding duplicate messages received by the compute node, and for each duplicate message, outputting an acknowledgement.

79. (new) Computer readable media embodying a program of instructions executable by a processor to perform a method of creating a checkpoint for a process in a distributed computation running on a distributed system, the process being performed by a compute node, the method comprising:

outputting an acknowledgement for each message received by the compute node;

creating a checkpoint for the process;

rolling the compute node back to the checkpoint in response to a failure; and

recognizing and discarding duplicate messages received by the compute node after the compute node is rolled back to the checkpoint, and for each duplicate message, outputting an acknowledgement.

80. (new) The computer readable media of claim 79 wherein the method further comprises storing in the memory a copy of each message output from the compute node until an acknowledgement is received, and outputting each message copied in the memory that does not receive an acknowledgement.

81. (new) Computer readable media embodying a program of instructions comprising a checkpoint library executable by a processor having access to an operating system and a distributed application to perform a process in a distributed computation running on a distributed system, the checkpoint library comprising:

instructions to create a checkpoint for the process, the creation of the checkpoint being transparent to the operating system and the distributed application.

82. (new) The computer readable media of claim 81 wherein the instructions to create the checkpoint comprise instructions to record a first set of memory locations modified by the process during a first checkpoint interval, and instructions to create the checkpoint from the contents of the first set of memory locations, while recording a second set of memory locations modified by the process during a second checkpoint interval.

83. (new) The computer readable media of claim 81 wherein the process is performed in a compute node in the distributed system, and wherein the instructions to create the checkpoint comprise instructions to store in memory a copy of each message output from the compute node during the process until an acknowledgement is received, and output each message copied in the memory that does not receive an acknowledgement, and instructions to receive messages during the process, output an acknowledgement for each message received, recognize and discard duplicate messages received by the compute node, and for each duplicate message, output an acknowledgement.

84. (new) Computer readable media embodying a program of instructions executable by a processor to perform a method of creating a checkpoint for a process in a distributed computation running on a distributed system, the method comprising:

creating a checkpoint for the process; and

storing the checkpoint to non-volatile memory and halting the process in response to a preemptive scheduling request.

85. (new) The computer readable media of claim 84 wherein the method further comprises using the stored checkpoint to resume the process.

86. (new) The computer readable media of claim 84 wherein the method further comprises performing a process in a second distributed computation running on the distributed system after the previous process is halted.

87. (new) The computer readable media of claim 84 wherein the process is performed in a compute node, the method further comprising performing a second process in the distributed computation previously being performed by a second compute node in the distributed system, the second process being resumed from the checkpoint last taken by the second compute node.

88. (new) A method of creating a checkpoint for a process in a distributed computation running on a distributed system, the method comprising:

recording a first set of memory locations modified by the process during a first checkpoint interval; and

creating a checkpoint from the contents of the first set of memory locations, while recording a second set of memory locations modified by the process during a second checkpoint interval.

89. (new) The method of claim 88 further comprising write protecting the first set of memory locations before the process modifies the second set of memory locations.

90. The method of claim 89 further comprising suspending the process between the first and second checkpoint intervals, and wherein the first set of memory locations are write protected while the process is suspended.

91. (new) The method of claim 90 further comprising entering a barrier following the completion of write protecting the first set of memory locations and exiting the barrier before resuming the process during the second checkpoint interval.

92. (new) The method of claim 89 wherein the checkpoint is created by storing the contents of the first set of memory locations, the method further comprising removing the write protection for a memory location from the first set when the process needs to modify the memory location during the second checkpoint interval after the contents of the memory location has been stored.

93. (new) The method of claim 92 wherein the checkpoint is created by storing the contents of the first set of memory locations in a certain order, the method further comprising storing the contents of a memory location from the first set earlier than it would otherwise be stored when the process needs to modify the memory location during the second checkpoint interval.

94. (new) The method of claim 93 further comprising removing the record of the memory location and the write protection for the memory location, after the contents of the memory location has been stored.

95. (new) The method of claim 88 wherein the checkpoint is created by storing the contents of the first set of memory locations to a checkpoint file.

96. (new) The method of claim 95 further comprising removing the record of a memory location from the first set after the contents from the memory location is stored in the checkpoint file.

97. (new) The method of claim 88 wherein the checkpoint is created by storing the contents of the first set of memory locations to non-volatile storage.

98. (new) The method of claim 88 wherein the process is performed by a compute node in the distributed system, the method further comprising storing in the memory a copy of each message output from the compute node during the process until an acknowledgement is received, and outputting each message copied in the memory that does not receive an acknowledgement, and wherein the method further comprises

receiving messages during the process, outputting an acknowledgement for each message received, recognizing and discarding duplicate messages received by the compute node, and for each duplicate message, outputting an acknowledgement.

99. (new) A method of creating a checkpoint for a process in a distributed computation running on a distributed system, the process being performed by a compute node, the method comprising:

storing in a memory a copy of each message output from the compute node until an acknowledgement is received;

creating a checkpoint for the process;

rolling compute node back to the checkpoint in response to a failure; and

outputting each message copied in the memory that does not receive an acknowledgement after the compute node is rolled back to the checkpoint.

100. (new) The method of claim 99 further comprising outputting an acknowledgement for each message received by the compute node, and recognizing and discarding duplicate messages received by the compute node, and for each duplicate message, outputting an acknowledgement.

101. (new) A method of creating a checkpoint for a process in a distributed computation running on a distributed system, the process being performed by a compute node, the method comprising:

outputting an acknowledgement for each message received by the compute node;

creating a checkpoint for the process;

rolling the compute node back to the checkpoint in response to a failure; and

recognizing and discarding duplicate messages received by the compute node after the compute node is rolled back to the checkpoint, and for each duplicate message, outputting an acknowledgement.

102. (new) The method of claim 101 further comprising storing in the memory a copy of each message output from the compute node until an acknowledgement is received, and outputting each message copied in the memory that does not receive an acknowledgement.

103. (new) A method of creating a checkpoint for a process in a distributed computation running on a distributed system, the method comprising:

creating a checkpoint for the process; and

storing the checkpoint to non-volatile memory and halting the process in response to a preemptive scheduling request.

104. (new) The method of claim 103 further comprising using the stored checkpoint to resume the process.

105. (new) The method of claim 103 further comprising performing a process in a second distributed computation running on the distributed system after the previous process is halted.

106. (new) The method of claim 103 wherein the process is performed in a compute node, the method further comprising performing a second process in the distributed computation previously being performed by a second compute node in the distributed system, the second process being resumed from the checkpoint last taken by the second compute node.

107. (new) A method of migrating a process in a distributed computation running on a distributed system from a first compute node to a second compute node in the distributed system, each of the first and second compute nodes having an operating system, the method comprising:

creating a checkpoint for the process in the first compute node; and migrating the process to the second compute node by providing the second compute node with the checkpoint without migrating the operating system from the first compute node to the second compute node.

108. (new) The method of claim 107 wherein the checkpoint is created by recording a first set of memory locations modified by the process in the first compute node during a first checkpoint interval, and creating the checkpoint from the contents of the first set of memory locations, while recording a second set of memory locations modified by the process in the first compute node during a second checkpoint interval.

109. (new) The method of claim 107 wherein the checkpoint is created by storing in memory a copy of each message output from the first compute node during the process in the first compute node until an acknowledgement is received, and outputting each message copied in the memory that does not receive an acknowledgement, and receiving messages during the process in the first compute node, outputting an acknowledgement for each message received, recognizing and discarding duplicate messages received by the first compute node, and for each duplicate message, outputting an acknowledgement.

REMARKS

Applicants respectfully request entry of the foregoing amendment prior to examination.

Please charge any additional fees which may be required, or credit overpayment to Deposit Account No. 50-1946.

Respectfully submitted,

5-16-06
Date

[illegible]
Craig A. Gelfound, Regis. No. 41,032

McDERMOTT WILL & EMERY, LLP
2049 Century Park East, 34th Floor
Los Angeles, CA 90067
Telephone: (310) 277-4110
Facsimile: (310) 277-4730